DOCUMENT RESUME

ED 033 044 SE 007 568

By-Gill, Gerald W.; Jensen, Alton P.

Management of Computer Programming. Part I: Practices and Problems.

Georgia Inst. of Tech., Atlanta. School of Information Sciences.
Spons Agency-National Science Foundation, Washington, D.C.
Report No-GITIS-69-01
Pub Date 69
Note-24p.
EDRS Price MF-$0.25 HC-$1.30
Descriptors-*Computer Programs, *Computers, *Computer Science, *Management, Management Development, *Program Administration
Identifiers-National Science Foundation

Investigated was the management of computer centers with emphasis on managing the programming effort. Problems and objectives of programming management were examined and techniques used in various business and governmental organizations are presented in the report. The aspects of the problem discussed in this report emphasize programming objectives and standards. The data for this investigation were obtained from a survey of seven organizations. A summary is presented on (1) the state of the art of managing the programming effort, and (2) projections for the future weighed against the problems, possibilities, and recommendations. (RP)

Work reported in this publication was performed at the School of Information Science, Georgia Institute of Technology. The primary support of this work came from the Georgia Institute of Technology and from agencies acknowledged in the report.

* * *

The graduate School of Information Science of the Georgia Institute of Technology offers comprehensive programs of education, research and service in the information, computer and systems sciences. As part of its research activities the School operates, under a grant from the National Science Foundation, an interdisciplinary science information research center. Correspondence concerning the programs and activities of the School may be addressed to Director, School of Information Science, Georgia Institute of Technology, Atlanta, Georgia 30332.

Telephone: (404) 873-4211

GITIS-69-01

(Research Report)

MANAGEMENT OF COMPUTER PROGRAMMING.
PART I: PRACTICES AND PROBLEMS

Gerald W. Gill

Alton P. Jensen

School of Information Science 1969

GEORGIA INSTITUTE OF TECHNOLOGY

Atlanta, Georgia

ACKNOWLEDGEMENT

The work reported in the paper has been sponsored in part by the National Science Foundation Grant GN-655. This assistance is gratefully acknowledged.

VLADIMIR SLAMECKA
Project Director

CONTENTS

ABSTRACT

This study investigates the management of computer centers, with emphasis on managing the programming effort. Problems and objectives of programming management are examined and techniques used in selected business and governmental organizations are presented.

The data were collected by the case study method by surveying seven organizations. The particular aspects of the problem discussed in this report emphasize programming objectives and standards.

This report is supplemented by the documentation of its case studies published in a separate volume (Gerald W. Gill and Alton P. Jensen, "Management of Computer Programming. Part II: Case Studies", Atlanta, Georgia, Georgia Institute of Technology, Jan. 1969).

INTRODUCTION

Since the computer became commonplace in industrial, governmental and public utility applications, technological change in hardware and software has outpaced the "humanware" required to control the machines and the specialists who make them work. Change has been rapid and continuous. The result has been a loss of control by management in the presence of unprecedented success.

The acute problems of managing the programming effort have been recognized since the middle of the last decade. But, because of the constant upward expansion of equipment and applications, managers have been too busy to think much about programming management.

In the early sixties, problems became so acute that management's attention was forced to the issue. Personnel problems include recruiting, training, evaluating and keeping qualified programmers. The programmer market is highly competitive and mobile. At any given time there are 50,000 more jobs than programmers.

Control problems include rating and controlling performance of programmers. Closely associated are the problems of program maintenance and protection against indispensable programmers.

The objective of "programming management" must be to organize the effort and establish personnel policies and standards to control methods and performance in order to solve these problems and effectively meet programming requirements.

This study has investigated the management of computer centers with emphasis on managing the programming effort. The report examines problems and objectives of programming management, and it presents techniques used in various business and governmental organizations.

PROBLEM AREAS IN PROGRAMMING MANAGEMENT

Organization of Computer Centers

Although it is popular to state that the computer system must be an independent department headed by a manager experienced in EDP, other arrangements can be successful. Other viewpoints hold that the computer should be under the department which will use it most or within the one with the highest clerical costs.

The manager of an MIS, data processing department or EDP department will find this decision already made for him. The point is, however, that as needs and applications change the organization will need revising.

In this particular investigation, EDP managers' backgrounds varied from a lawyer to an accountant. This is not to discount the advisability of an appropriate background. The real lesson is that organization and management are variable from company to company and from individual to individual.

To establish some common ground, the following organization chart is presented. It is not offered as correct or as a recommended structure, but as one basic grouping of tasks necessary to most computer centers. The functions are not discussed at this time as seven particular organizations will be examined later.

Basic to the purposes of this report are the decisions as to how programmers are organized. Programmers and Systems Analysts can be separated or one group. Programmers may be organized as a team or as functional groups. Also important is how the organization provides for control.

Personnel

1. Recruiting. The soaring cost of programming, delays in job completion and debugging, and the cost of replacing programmers, demands more than Parkinson's "Reject everyone over 50, under 20 and anyone called Murphy."

Sources include other company's programmers, employees of your own organization, inexperienced but schooled graduates of recognized universities, technical schools, business schools and programming institutes.

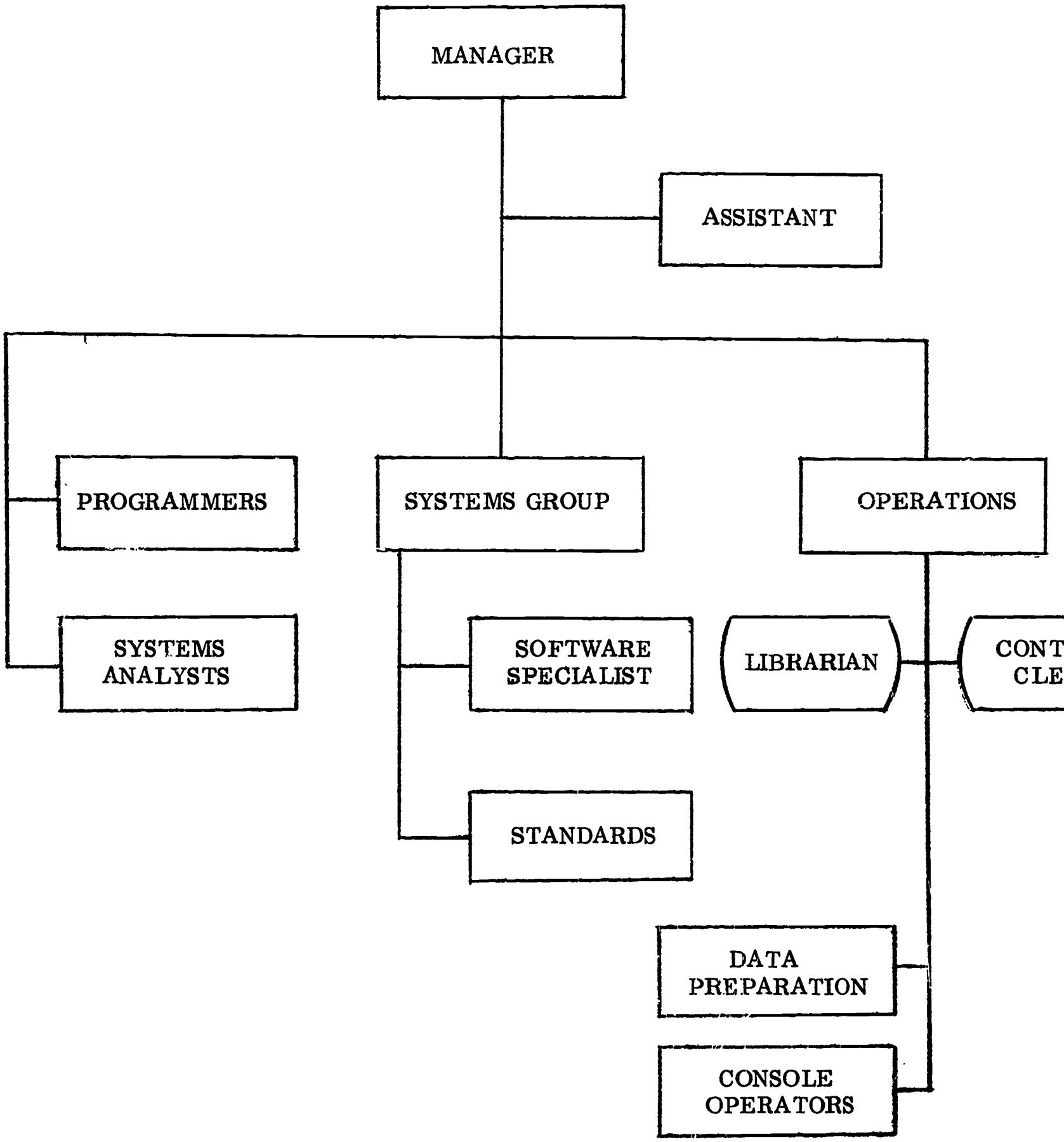

Fig. 1. A Sample Organization Chart for a Computer Center

Screening is usually by application, interview and testing. Many tests are available. The most popular is the EDPM aptitude test prepared by the Psychological Corporation in 1955, which is usually called the IBM PAT. Some controversy exists over the fact that this test is timed. Brandon Enterprises offers an aptitude test de-emphasizing speed.

2. Training. This varies, of course, with the experience of the trainee. All manufacturers provide a wide variety of instructional programs both formally and informally. Frequently such courses are administered parallel to on-the-job training.

3. Retaining. The tendency has been to compete financially with other companies to retain valuable programmers by outbidding competitors. There are several dangers in such practices. Other factors affecting retention are advancement opportunity, (e.g., operator - programmer - systems analysts - manager) challenging work, loyalty, pleasant working situation and personal relations.

4. Evaluating. This has to be largely subjective for programmers. Close observation helps. You can watch a programmer, but perhaps more can also be learned by watching his associates work with him. Observable rating factors include:

- Viewpoint (systems or narrow)
- Promotability
- Loyalty
- Proper documentation
- Test routines and debugging
- Meeting of reasonable deadlines.

Control vs. Change

Management must protect itself against errors, problems of personnel turnover, indispensable programmers and fraud or sabotage. This calls for control over information and control over people. Brandon distinguishes between control over performance and control over methods, prescribing standards as the first step toward achieving management control.

Few things are as prone to change as information systems. Change introduces disorder and can cause loss of control. An information system evolves and with every change it evolves more and faster irreversibly. Just as a home run cannot be undone

in a ball game, once a system changes, the job, people and environment are changed forever.

Control costs money. Therefore, a basic question is: how much control is required? Some elements of control are educational and psychological. However, in modern computer uses, measurable techniques must be employed when possible. Scientific methods are available. Such methods are not used as effectively or as extensively as they could be.

The goal cannot be to maintain the "status quo". Managers must retain the control necessary to direct away from confusion and destruction toward a state of order which will survive change. An example of this kind of thinking would be the attitude that a program begins when it goes on the air.

A few comments are given on the subject of controls.

— Control methods must assure the security and quality of information processing.

— The best controls are active and prevent errors from entering the system to explode throughout.

— The tendency is to overcontrol or not control at all.

— Controls should be used with moderation.

— Statistics should be kept and monitored.

— A separate group of people must exist within the organization to establish and administer control.

OBJECTIVES AND STANDARDS

Objectives

Now that the problems have been discussed the objective will be expanded to include establishing and maintaining—

1. The organizational structure and job definitions to minimize duplication of effort and maximize effectiveness.

2. Personnel policies that will successfully locate, train, evaluate, retain and control qualified programmers.

3. Control and protection against error and change.

4. Efficient methods of scheduling and forecasting programming projects.

5. Effective program maintenance.

6. Constant review of progress and a flexibility which will prevent loss of control with future changes.

Brandon explains that installation of ADP equipment is the most technical task management has ever faced. As a result, there has been little organization or control applied. Thus we have a self-perpetuating condition of questionable efficiency and economy.

Hopefully the methods of programming management should reduce costly programming waste, personnel turnover and ineffectual performance on equipment.

There are tools for these objectives, but first a caution that a tool can be bad, good or improperly used. For instance, a bad standard might add time and expense instead of helping.

Standards

1. Standards and Management. Planning and control are difficult in this infant field. A case can be built against the use of standards, but most managers deem them necessary as the medium for management control. Methods standards and performance standards are linked through documentation.

A common problem is that everyone believes in standards -- for others. A frequent complaint is that standards restrict creativity and increase workloads. But Brandon dismisses this as a myth.

On the other hand there is little doubt that the use of good standards aids planning, increases control, enables management to measure and predict performance, simplifies training, improves communication and eases conversions.

Standards must have the support of top management and must be based upon experience. The objective must be practicability and not just a requirement for "paperwork". In the form of job descriptions and terminology, standards should minimize confusion. In the form of work documentation and program documentation, good standards improve communications.

Performance standards should improve management's ability to evaluate and predict. If good standards exist, planning for programming can be improved. Records can be kept and analyzed to enable consideration of the complexity of the job, experience of programmer, similarity to previous programs, program language to be used, standard routines available, etc. These records must be further refined by adding time taken for training courses, other duties, illness and vacation. Surely the guesswork will be less haphazard if these areas are standardized to allow estimates for such factors.

The prime question for management is whether standards can be enforced. This involves personalities, attitudes and management techniques. The end result must be an adjusting of the level of detail to a reasonable level which is to be strictly enforced.

2. Standards and Programmers. The tendency among programmers is to resist being pinned down with detailed job specifications. As programming progresses, ways are usually discovered to save time, save space, or just be more sophisticated.

A dictionary definition reflects that standards are "authoritative models or measures, or patterns for guidance taken by general consent as a basis of comparison." This implies that without "general consent" you have no standards. Thus it is important to communicate the need for standards to programmers and to involve them in creating and updating standards.

For example, a standard method of programming and documenting should make program maintenance and reprogramming for conversions easier. This should be a selling point to programmers. Furthermore, standards should mean that when a program is finished the programmer will not be continually questioned by operators, supervisors, users and librarians.

When large programs are created in modules and tested individually, standard structures are the only suitable means of communications between various programmers. Files and tape libraries must be indexed, cross referenced and made available to programmers.

Any work is simplified if it is clearly defined and nothing is worse than not knowing exactly what is expected. Thus standard formats, preferred practices, meaningful names and symbols and a definite reference for questions concerning these practices can become pleasant for all concerned.

3. National Standards. There is a need for comprehensive national standards in some programming areas. Few exist! Nor can more be expected in the near future. Standards differ with machines, types of machines, manufacturer, industry, user and department. The result is that each department must standardize within.

Meanwhile, government organizations such as ASA and ISO and trade organizations such as ACM and DPMA continue to try. The number one offenders are the manufacturers who change terminology with changes in technology.

4. Implementing Standards. Certain standards are inherited. That is, standards are imposed by the computer or by the language used. Even if standards are accepted, they must be communicated. Manufacturers' manuals outline the hardware and language limitations.

But, within each organization, standard procedures must be in writing and enforced. This must include specifications, progress and costs reporting, formal change procedures, good programming practices and documentation. Documentation refers to clearly outlined work statements, system design, program design, data descriptions and testing results. But standards themselves must be documented. The most important instrument for this is usually called the "Standards Manual".

The Standards Manual

Initially it must be decided whether this is to be a reference or a working manual. It is doubtful that it can be both. Also, there is no need to compete with the manufacturer by duplicating portions of his manuals on hardware or language.

This manual should be readily available and easy to use. Some simple prescribed method of indexing and posting changes is required. A beautiful book adds decor but little else if it is difficult to use. People trust their memory rather than to look up

the details. One popular format is a loose-leaf binder with various sections as needed. A revision log reflecting replaced pages and dates simplifies the job of keeping the manual current. Each manual should have one person responsible for its currency.

This manual should be kept as brief as possible. For instance if there are 200 subroutines to describe, this might make the book so thick that it would be frightening. Possibly these could be stored separately for reference leaving a thinner and more friendly looking manual.

Inside the manual, there should be clear, direct, concise and simple directions for its use. Wasted pages and words should be absent and the tone should be informal. In short, it should be written to express instead of to impress!

Each manufacturer will provide assistance in establishing a standards manual. For example, IBM provides a manual C20-1670-0, Management Planning Guide for a Manual of Data Processing Standards. This manual is comprehensive yet detailed and specific. It should be helpful in most cases, however, each particular organization must tailor-fit a manual to its own needs. Minimum contents for most any manual would include sections on—

I Systems Analysis
II Systems Documentation
III Program Documentation
IV Production Run Procedures
V Controls
VI Standard Macros
VII Structure and Functions within Department
VIII Miscellaneous
IX Revision Log

Documentation

Documentations should include description of data elements, layouts, file specifications and applications descriptions by system. A documentation dossier for each program includes flow charts relating the program to the system. Further it contains a program flow chart, a narrative explanation, layouts, run procedures, a listing with sample data and a complete record of revisions.

Documentation also includes the use of standard forms, decision tables, codes, operating dossiers, and schedules. To repeat a previous statement, the standards manual itself is a document describing required procedures.

Controls

All cards, tapes, disk files, dossiers, forms, handbooks, manuals, and programs must be logged, coded, filed, indexed, backed-up, safeguarded and revised. Schedules must be created and maintained. Networks such as PERT and GANT charts promise some help in scheduling.

A particular weakness is in systems analysis and problem specification to programmers. Problems, programs, and people are hard to time or evaluate. Rating performance is basically a matter of experience. Speed is not the only factor in programming as run time, memory space, maintainability, documentation, and team work are other factors. Standard forms and documents increase control in this respect. Program documentation packages, production packages, systems packages and program specifications strengthen control over both methods and performances. Thus documentation should be modeled as a data bank and provide control for the analysis, programming and production stages.

A good practice is to employ at least two programmers on any job, preferably all the way through testing. This should optimize techniques, educate all members of the team and minimize errors. Also, with more people familiar with the program, there is some protection against programmer turnover problems.

It can be difficult to control programmers if they work in one big group physically located in one large room. Private offices seem to be preferred, but are too expensive. The common physical arrangement is two programmers per office.

CURRENT STATUS OF PROGRAMMING MANAGEMENT

The results of this study are based on a survey of seven organizations. Actual survey data are published in a complementary, separate report (G. W. Gill and A. P. Jensen, "Management of Computer Programming. Part II: Case Studies." Atlanta, Ga., Georgia Institute of Technology, Jan. 1969).

The seven participating organizations represented the following areas and operated the equipment indicated:

1. A Public Utility Company; IBM 360/40 with IBM 1287 Optical Scanner and Data Cell

2. A Manufacturing and Distribution Company; IBM 360/65 with Disk and Associated Communications Equipment

3. A Local Government Agency; IBM 360/30, 20

4. A Heavy Manufacturing Company; IBM 360/30 and a special purpose process control computer

5. An Aerospace Contracting Company;IBM 360/50, 30 with a Philco 6000 Multi-Font Print Reader and a wide range information processing devices.

6. A Large Federal Government Agency;UNIVAC 1108's, IBM 7094's, 360's, etc.

7. A Software Contracting and Consulting Company; no equipment in house.

Findings of the Survey

The following is a summary of the state of the art of managing the programming effort, as determined by the survey.

1. The structural trend is to locate the computer within a separate staff reporting directly to the president of the organization.

2. Little uniformity is found in programmer organization, except that no company divided its programmers into groups such as flow charters and coders. Further there is no trend toward separate program maintenance groups.

3. All companies are pleased with the support given by manufacturers -- now. There have been past problems and two managers expressed mild displeasure concerning software.

4. All organizations were familiar with the problems investigated — many by first-hand experience. Most have learned that one failure is not terminal.

5. Indications are that, although the demand still exceeds the supply of programmers, the prime targets are now inexperienced persons from within and new graduates of respected schools. Two years experience is no longer a prerequisite.

6. Training is done on the job and is supplemented by the training programs of manufacturers.

7. Conversion of equipment, software and applications is the normal state of affairs.

8. Changes are usually under intense pressure which often relegate standards and documentation to low priorities. All managers have standards and require documentation. A definite gap exists between desires and practices.

9. No results are evident from efforts of national standards associations. No standard language is in sight. (There was one dissenting opinion on PL/1.) There is little hope for compatability between machines, languages and programs -- soon. Everyone is resigned to live forever with non-standard terminology. Not one glossary was found at any level.

10. Standards and control for systems analysis and problem specification are weak to non-existent. Programming could be described as adolescent and systems analysis as infantile.

11. Controls are weak - especially in areas involving manual forms. Management is committed to blind trust in the loyalty, honesty and ability of people. Planning and scheduling is largely based on fear and superstititon rather than scientific skill and requirements.

12. Optimism pervades. Ideas are being formulated, exchanged, and tried.

Projections for the Future

1. Problems. The excessive demand for programmers today and the "sweat shop" environment resulting from long hours and pressure may birth new problems. There are faint rumblings of programming fraud. This could grow into a problem and

be joined with sabotage. More serious could be a mass influx of non-dedicated and non-professional programmers working beside experienced programmers who have programmed for 25 years, experienced 20 conversions, have not been promoted, and have lost enthusiasm. The result: "old burned-out programmers" corrupting and limiting the enthusiasm of youth.

2. Possibilities. There are many areas of promise! Some national standards may come to pass. Some manufacturers may strive for compatability. Perhaps major software "break-throughs" are in store. Wired-in programs offer hope with the "compiler in a box" concept. Scientific methods for planning and scheduling are possible. For example, if graphs could be constructed weighing such factors as program length, type, level of complexity, previous patches, needed changes and type of changes, age of program, etc., it is conceivable that from the resulting graphs could be determined the desirability of complete reprogramming over program maintenance. Similar tools are possible for scheduling, evaluating, etc. But they would have to be non-frightening and easy to use and accepted by managers.

3. Recommendations. College students should be allowed past the in-out counter to get hands-on experience and to study operations...not for the purpose of programming, but to facilitate management and understanding. Programming courses should present documentation standards as an integral part of programming. Courses should provide contact with actual installations and real people.

Users are neglecting college students as a promising source of programmers. The Co-op plan is outstanding in that students simultaneously receive schooling with professional practice. If these students are employed in pairs, one is always on the job. In six years, the company has two graduates with three years experience. Other students could be used part time or as part of course work.

It is ironic that the possessors of the world's most sophisticated information systems are themselves weak in the flow of information. First of all they must clearly express and demand what they want of the manufacturers, especially with respect to software. Secondly, a free exchange of ideas between non-competitive organizations would help solve problems and provide new ideas in programming management. The requirement is for a concerted, cooperative, and coordinated effort by responsible managements to place programming in a management perspective. Accomplishing this without sacrificing the creativity and productivity of the novice, professional, or virtuoso programmer is the challenge.

BIBLIOGRAPHY

Awad, Elias M., Business Data Processing. Prentice-Hall, Inc., Englewood Cliffs, N. J. 1965.

Brandon, Dick H., Management Standards for Data Processing. D. Van Nostrand Company, Princeton, N.J. 1963.

Canning, Richard G., Managing the Programming Effort. "The EDP Analyzer". Canning Publications, Inc. Vista, Calif. June 1968.

Davis, Gordon B., An Introduction to Electronic Computers. McGraw-Hill, New York, 1965.

Drucker, Peter F., The Effective Executive. Harper & Row, New York, 1967.

Management Planning Guide for a Manual of Data Processing Standards. IBM. C20-1670-0, Second Edition

Parkinson, C. Northcote, Parkinson's Law. Houghton Mifflin Co., Boston. 1957

Rothery, Brian, "Programming and Irreversibility". Data Processing Magazine. North American Publishing Company, Philadelphia, Pa. Dec. 1967.

Rothery, Brian, "The Problems of Program Change", Data Processing Magazine. North American Publishing Co., Phila., Pa. March 1968.

Smith, William D., Computer Programming Schools Offer Opportunity. Business and Finance Section "The New York Times", N.Y., Sept. 17, 1967.

Techt, Charles Philip, The Management of Programming Projects. American Management Association, Inc. 1968.

Wilder, Neal, Two Years Experience: Is it a Valid Criterion? "Computerworld" Boston, Mass. July 17, 1968.

Wolf, Jack M., Ph.D., Aptitude Assessment Battery Programming. Brandon/ Systems Press, New York, 1968.

APPENDIX

A MANAGEMENT POLICY STATEMENT

I. INTRODUCTION

In order to assure that project requirements are met in a timely, economical, efficient and accurate manner, the following project policy and implementation procedures are to be followed.

II. PHILOSOPHY

A. System Design

The over-all system design and definition of a computer application is usually a complex task requiring a high level of logical thought and detailed planning. Such logical planning is performed more easily, more timely, and with greater sophistication if more than one analyst is involved in the initial design. The early design phase of a product determines its ultimate quality. The most experienced computer professionals should be used during this phase.

B. Programming

In computer programming, one hundred per cent accuracy is an absolute objective. All steps must be taken to assure that not one single error remains in the final product. Careful planning, coding and checking must be followed by test runs under all conceivable conditions before a program can be considered operational.

C. Documentation

The most efficient and sophisticated computer system application is useless without complete, easily understood documentation including system description, flow charts, program listings, data formats, output format descriptions and run instructions. Such documentation is a necessity if someone other than one of the original program authors is to make the inevitable changes and modifications necessary to satisfy new or unanticipated requirements on the system.

D. Operation

An application is not completed until the system is operating under its "permanent" operating personnel. Training or orientation required by operating personnel

in order to use the system effectively must be provided. Proper staffing at the time of initial operations is essential to the success of any project.

III. IMPLEMENTATION

Implementation of the principals set forth in the previous section can be accomplished according to the task outline set forth below. A schedule is established which estimates the time spent on each phase of the project. As the work progresses, the project leader must keep informed of the schedules and financial status. When each increment of 25 per cent of the estimated total project cost is reached, higher management and the project leader should review status and revise estimated completion time and cost. The work proceeds according to the following outline:

1. Analysis of the subject systems requirements to identify all input data and establish the general format for necessary and desired reports. Establish the run frequency and distribution of such reports.
2. Analysis of the present mode of operation to determine what changes should be made to the system to facilitate an automated approach.
3. Design and documentation of the total system "in-the-large" including macro flow charts, relationship of individual modules and existing library packages as well as complete descriptions and samples of all input and output data.
4. The proposed system must be reviewed by the operations departments, consultants, and systems designers to ensure that all requirements have been met, that all data formats are acceptable and that the total system design is logically complete and sound. Modifications to take care of any discrepancies or desired features should be made at this time. This review should include refined cost analysis and updated time schedules.
5. System programming, coding, and testing under the supervision of the project leader should ensure a high quality product. All work must be checked and re-checked including comparison of test run

cases against known results. All documentation must be reviewed by an analyst, not a participating author, to determine that it is complete and readily understood.

6. Management approval may be obtained after a staff consultant reviews the completed work package and certifies that it is ready for use.
7. Delivery and installation of final equipment and software is supervised by the project leader who arranges for any operating personnel training that may be necessary. All test cases should be re-run to ensure that the final operating system produces the same results as the test system.
8. Before final acceptance, the entire system will be in operation for a period of time sufficient to identify faults or desired modifications.